TRACKING SEMANTIC OBJECTS IN VECTOR IMAGE SEQUENCES

FIELD OF THE INVENTION

The invention relates to analysis of video data, and more, specifically relates to a method for tracking meaningful entities called semantic objects as they move through a sequence of vector images such as a video sequence.

BACKGROUND OF THE INVENTION

A semantic video object represents a meaningful entity in a digital video clip, e.g., a ball, car, plane, building, cell, eye, lip, hand, head, body, etc. The term "semantic" in this context means that the viewer of the video clip attaches some semantic meaning to the object. For example, each of the objects listed above represent some real-world entity, and the viewer associates the portions of the screen corresponding to these entities with the meaningful objects that they depict. Semantic video objects can be very useful in a variety of new digital video applications including content-based video communication, multimedia signal processing, digital video libraries, digital movie studios, and computer vision and pattern recognition. In order to use semantic video objects in these applications, object segmentation and tracking methods are needed to identify the objects in each of the video frames.

The process of segmenting a video object refers generally to automated or semi-automated methods for extracting objects of interest in image data. Extracting a semantic video object from a video clip has remained a challenging task for many years. In a typical video clip, the semantic objects may include disconnected components, different colors, and multiple rigid/non-rigid motions. While semantic objects are easy for viewers to discern, the wide variety of shapes, colors and motion of semantic objects make it difficult to automate this process on a computer. Satisfactory results can be achieved by having the user draw an initial outline of a semantic object in an initial frame, and then use the outline to compute pixels that are part of the object in that frame. In each successive frame, motion estimation can be used to predict the initial boundary of an object based on the segmented object from

the previous frame. This semi-automatic object segmentation and tracking method is described in co-pending U.S. Patent Application No. 09/054,280, by Chuang Gu, and Ming Chieh Lee, entitled Semantic Video Object Segmentation and Tracking, which is hereby incorporated by reference.

Object tracking is the process of computing an object's position as it moves from frame to frame. In order to deal with more general semantic video objects, the object tracking method must be able to deal with objects that contain disconnected components and multiple non-rigid motions. While a great deal of research has focused on object tracking, existing methods still do not accurately track objects having multiple components with non-rigid motion.

Some tracking techniques use homogeneous gray scale/color as a criterion to track regions. See F. Meyer and P. Bouthemy, "Region-based tracking in an image sequence", ECCV'92, pp. 476-484, Santa Margherita, Italy, May 1992; Ph. Salembier, L. Torres, F. Meyer and C. Gu, "Region-based video coding using mathematical morphology", Proceeding of the IEEE, Vol. 83, No. 6, pp. 843-857, June 1995; F. Marques and Cristina Molina, "Object tracking for content-based functionalities", VCIP'97, Vol. 3024, No. 1, pp. 190-199, San Jose, Feb., 1997; and C. Toklu, A. Tekalp and A. Erdem, "Simultaneous alpha map generation and 2-D mesh tracking for multimedia applications", ICIP'97, Vol. I, page 113-116, Oct., 1997, Santa Barbara.

Some employ homogenous motion information to track moving objects. See for example, J. Wang and E. Adelson, "Representing moving images with layers", IEEE Trans. on Image Processing, Vol. 3, No. 5. pp. 625-638, Sept. 1994 and N. Brady and N. O'Connor, "Object detection and tracking using an em-based motion estimation and segmentation framework", ICIP'96, Vol. I, pp. 925-928, Lausanne, Switzerland, Sept. 1996.

Others use a combination of spatial and temporal criteria to track objects. See M.J. Black, "Combining intensity and motion for incremental segmentation and tracking over long image sequences", ECCV'92, pp. 485-493, Santa Margherita, Italy, May 1992; C. Gu, T. Ebrahimi and M. Kunt, "Morphological moving object segmentation and tracking for content-based video coding", Multimedia

Communication and Video Coding, pp. 233-240, Plenum Press, New York, 1995; F. Moscheni, F. Dufaux and M. Kunt, "Object tracking based on temporal and spatial information", in Proc. ICASSP'96, Vol. 4, pp. 1914-1917, Atlanta, GA, May 1996; and C. Gu and M.C. Lee, "Semantic video object segmentation and tracking using mathematical morphology and perspective motion model", ICIP'97, Vol. II, pages 514 – 517, Oct. 1997, Santa Barbara.

Most of these techniques employ a forward tracking mechanism that projects the previous regions/objects to the current frame and somehow assembles/adjusts the projected regions/objects in the current frame. The major drawback of these forward techniques lies in the difficulty of either assembling/adjusting the projected regions in the current frame or dealing with multiple non-rigid motions. In many of these cases, uncertain holes may appear or the resulting boundaries may become distorted.

Figures 1A-C provide simple examples of semantic video objects to show the difficulties associated with object tracking. Figure 1A shows a semantic video object of a building 100 containing multiple colors 102, 104. Methods that assume that objects have a homogenous color do not track these types of objects well. Figure 1B shows the same building object of Figure 1A, except that it is split into disconnected components 106, 108 by a tree 110 that partially occludes it. Methods that assume that objects are formed of connected groups of pixels do not track these types of disconnected objects well. Finally, Figure 1C illustrates a simple semantic video object depicting a person 112. Even this simple object has multiple components 114, 116, 118, 120 with different motion. Methods that assume an object has homogenous motion do not track these types of objects well. In general, a semantic video object may have disconnected components, multiple colors, multiple motions, and arbitrary shapes.

In addition to dealing with all of these attributes of general semantic video objects, a tracking method must also achieve an acceptable level of accuracy to avoid propagating errors from frame to frame. Since object tracking methods typically partition each frame based on a previous frame's partition, errors in the previous frame tend to get propagated to the next frame. Unless the tracking method computes an

object's boundary with pixel-wise accuracy, it will likely propagate significant errors to the next frame. As result, the object boundaries computed for each frame are not precise, and the objects can be lost after several frames of tracking.

SUMMARY OF THE INVENTION

The invention provides a method for tracking semantic objects in vector image sequences. The invention is particularly well suited for tracking semantic video objects in digital video clips, but can also be used for a variety of other vector image sequences. While the method is implemented in software program modules, it can also be implemented in digital hardware logic or in a combination of hardware and software components.

The method tracks semantic objects in an image sequence by segmenting regions from a frame and then projecting the segmented regions into a target frame where a semantic object boundary or boundaries are already known. The projected regions are classified as forming part of a semantic object by determining the extent to which they overlap with a semantic object in the target frame. For example, in a typical application, the tracking method repeats for each frame, classifying regions by projecting them into the previous frame in which the semantic object boundaries are previously computed.

The tracking method assumes that semantic objects are already identified in the initial frame. To get the initial boundaries of a semantic object, a semantic object segmentation method may be used to identify the boundary of the semantic object in an initial frame.

After the initial frame, the tracking method operates on the segmentation results of the previous frame and the current and previous image frames. For each frame in a sequence, a region extractor segments homogenous regions from the frame. A motion estimator then performs region based matching for each of these regions to identify the most closely matching region of image values in the previous frame. Using the motion parameters derived in this step, the segmented regions are projected into the previous frame where the semantic boundary is already computed. A region

classifier then classifies the regions as being part of semantic object in the current frame based on the extent to which the projected regions overlap semantic objects in the previous frame.

The above approach is particularly suited for operating on an ordered sequence of frames. In these types of applications, the segmentation results of the previous frame are used to classify the regions extracted from the next frame. However, it can also be used to track semantic objects between an input frame and any other target frame where the semantic object boundaries are known.

One implementation of the method employs a unique spatial segmentation method. In particular, this spatial segmentation method is a region growing process where image points are added to the region as long as the difference between the minimum and maximum image values for points in the region are below a threshold. This method is implemented as a sequential segmentation method that starts with a first region at one starting point, and sequentially forms regions one after the other using the same test to identify homogenous groups of image points.

Implementations of the method include other features to improve the accuracy of the tracking method. For example, the tracking method preferably includes region-based preprocessing to remove image errors without blurring object boundaries, and post-processing on the computed semantic object boundaries. The computed boundary of an object is formed from the individual regions that are classified as being associated with the same semantic object in the target frame. In one implementation, a post processor smooths the boundary of a semantic object using a majority operator filter. This filter examines neighboring image points for each point in a frame and determines the semantic object that contains the maximum number of these points. It then assigns the point to the semantic object containing the maximum number of points.

Further advantages and features of the invention will become apparent in the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figs. 1A-C are examples illustrating different types of semantic objects to illustrate the difficulty of tracking general semantic objects.

Fig. 2 is a block diagram illustrating a semantic object tracking system.

Figs. 3A-D are diagrams illustrating examples of partition images and a method for representing partition images in a region adjacency graph.

Fig. 4 is a flow diagram illustrating an implementation of a semantic object tracking system.

Fig. 5 is a block diagram of a computer system that serves as an operating environment for an implementation of the invention.

DETAILED DESCRIPTION

Overview of a Semantic Object Tracking System

The following sections describe a semantic object tracking method. This method assumes that the semantic object for the initial frame (I-frame) is already known. The goal of the method is to find the semantic partition image in the current frame based on the information from the previous semantic partition image and the previous frame.

One fundamental observation about the semantic partition image is that the boundaries of the partition image are located at the physical edges of a meaningful entity. A physical edge is the position between two connected points where the image value (e.g., a color intensity triplet, gray scale value, motion vector, etc.) at these points are significantly different. Taking advantage of this observation, the tracking method solves the semantic video object tracking method using a divide-and-conquer strategy.

First, the tracking method finds the physical edges in the current frame. This is realized using a segmentation method, and in particular, a spatial segmentation method. The goal of this segmentation method is to extract all the connected regions with homogeneous image values (e.g., color intensity triplets, gray scale values, etc.) in the current frame. Second, the tracking method classifies each extracted region in

the current frame, to determine which object in the previous frame it belongs to. This classification analysis is a region-based classification problem. Once the region-based classification problem is solved, the semantic video object in the current frame has been extracted and tracked.

Figure 2 is a diagram illustrating the semantic video object tracking system. The tracking system comprises the following five modules:

1. region pre-processing 220;
2. region extraction 222;
3. region based motion estimation 224;
4. region-based classification 226; and
5. region post-processing 228.

Figure 2 uses the following notation:

$\mathbf{I}_i$ - input image for frame i;

$\mathbf{S}_i$ - spatial segmentation results for frame i;

$\mathbf{M}_i$ - motion parameters for frame i; and

$\mathbf{T}_i$ - tracking results for frame i.

The tracking method assumes that the semantic video object for the initial frame $\mathbf{I}_0$ is already known. Starting with an initial frame, a segmentation process determines an initial partition defining boundaries of semantic objects in the frame. In Fig. 2, the I-segmentation block 210 represents a program for segmenting a semantic video object. This program takes the initial frame $\mathbf{I}_0$ and computes the boundary of a semantic object. Typically, this boundary is represented as a binary or alpha mask. A variety of segmentation approaches may be used to find the semantic video object(s) for the first frame.

As described in co-pending U.S. Patent Application No. 09/054,280 by Gu and Lee, one approach is to provide a drawing tool that enables a user to draw a border around the inside and outside of a semantic video object's boundary. This user-drawn boundary then serves as a starting point for an automated method for snapping the

computed boundary to the edge of the semantic video object. In applications involving more than one semantic video object of interest, the I-segmentation process 210 computes a partition image, e.g., a mask, for each one.

The post-processing block 212 used in the initial frame is a process for smoothing the initial partition image and for removing errors. This process is the same or similar to post-processing used to process the result of tracking the semantic video object in subsequent frames, $\mathbf{I}_1$, $\mathbf{I}_2$.

The input for the tracking process starting in the next frame ($\mathbf{I}_1$) includes the previous frame $\mathbf{I}_0$ and the previous frames segmentation results $\mathbf{T}_0$. The dashed lines 216 separate the processing for each frame. Dashed line 214 separates the processing for the initial frame and the next frame, while dashed line 216 separates the processing for subsequent frames during the semantic video object tracking frames.

Semantic video object tracking begins with frame $\mathbf{I}_1$. The first step is to simplify the input frame $\mathbf{I}_1$. In Fig. 2, simplification block 220 represents a region-preprocessing step used to simplify the input frame $\mathbf{I}_1$ before further analysis. In many cases, the input data contains noise that may adversely effect the tracking results. Region-preprocessing removes noise and ensures that further semantic object tracking is carried out on the cleaned input data.

The simplification block 220 provides a cleaned result that enables a segmentation method to extract regions of connected pixels more accurately. In Fig. 2, the segmentation block 222 represents a spatial segmentation method for extracting connected regions with homogeneous image values in the input frame.

For each region, the tracking system determines whether a connected region originates from the previous semantic video object. When the tracking phase is complete for the current frame, the boundary of the semantic video object in the current frame is constructed from the boundaries of these connected regions. Therefore, the spatial segmentation should provide a dependable segmentation result for the current frame, i.e., no region should be missing and no region should contain any area that does not belong to it.

The first step in determining whether a connected region belongs to the semantic video object is matching the connected region with a corresponding region in the previous frame. As shown in Fig. 2, a motion estimation block 226 takes the connected regions and the current and previous frames as input and finds a corresponding region in the previous frame that most closely matches each region in the current frame. For each region, the motion estimation block 226 provides the motion information to predict where each region in the current frame comes from in the previous frame. This motion information indicates the location of each region's ancestor in the previous frame. Later, this location information is used to decide whether the current region belongs to the semantic video object or not.

Next, the tracking system classifies each region as to whether it originates from the semantic video object. In Fig. 2, the classification block 226 identifies the semantic object in the previous frame that each region is likely to originate from. The classification process uses the motion information for each region to predict where the region came from in the previous frame. By comparing the predicted region with the segmentation result of the previous frame, the classification process determines the extent to which the predicted region overlaps a semantic object or objects already computed for the previous frame. The result of the classification process associates each region in the current frame either with a semantic video object or the background. A tracked semantic video object in the current frame comprises the union of all the regions linked with a corresponding semantic video object in the previous frame.

Finally, the tracking system post-processes the linked regions for each object. In Fig. 2, post processing block 228 fine tunes the obtained boundaries of each semantic video object in the current image. This process removes errors introduced in the classification procedure and smoothes the boundaries to improve the visual effect.

For each subsequent frame, the tracking system repeats the same steps in an automated fashion using the previous frame, the tracking result of the previous frame and the current frame as input. Fig. 2 shows an example of the processing steps repeated for frame $\mathbf{I}_2$. Blocks 240-248 represent the tracking system steps applied to the next frame.

Unlike other region and object tracking systems that employ various forward tracking mechanisms, the tracking system shown in Fig. 2 performs backward tracking. The backward region-based classification approach has the advantage that the final semantic video object boundaries will always be positioned in the physical edges of a meaningful entity as a result of the spatial segmentation. Also, since each region is treated individually, the tracking system can easily deal with disconnected semantic video objects or non-rigid motions.

Definitions

Before describing an implementation of the tracking system, it is helpful to begin with a series of definitions used throughout the rest of the description. These definitions help illustrate that the tracking method applies not only to sequences of color video frames, but to other temporal sequences of multi-dimensional image data. In this context, "multi-dimensional" refers to the spatial coordinates of each discrete image point, as well as the image value at that point. A temporal sequence of image data can be referred to as a "vector image sequence" because it consists of successive frames of multi-dimensional data arrays. As an example of a vector image sequence, consider the examples listed in Table 1 below:

Vector image	Dimensions	Explanation
$I_t: (x, y) \rightarrow Y$	n = 2, m = 1	gray-tone image sequence
$I_t: (x, y) \rightarrow (V_x, V_y)$	n = 2, m = 2	dense motion vector sequence
$I_t: (x, y) \rightarrow (R, G, B)$	n = 2, m = 3	color image sequence
$I_t: (x, y, z) \rightarrow Y$	n = 3, m = 1	gray-tone volume image sequence
$I_t: (x, y, z) \rightarrow (V_x, V_y)$	n = 3, m = 2	dense motion vector volume sequence
$I_t: (x, y, z) \rightarrow (R, G, B)$	n = 3, m = 3	color volume image sequence

Table 1. Several types of input data as a vector image sequences

The dimension, *n*, refers to the number of dimensions in the spatial coordinates of an image sample. The dimension, *m*, refers to the number of dimensions of the image value located at the spatial coordinates of the image sample. For example, the spatial coordinates of a color volume image sequence include three spatial coordinates defining the location of an image sample in three-dimensional space, so n = 3. Each sample in the color volume image has three color values, R, G, and B, so m = 3.

The following definitions provide a foundation for describing the tracking system in the context of vector image sequences using set and graph theory notation.

Definition 1 Connected points:

Let **S** be a *n*-dimensional set: a point $p \in \mathbf{S} \Rightarrow p = (p_1, \ldots, p_n)$. $\forall p, q \in \mathbf{S}, p$ and q are connected if and only if their distance $D_{p,q}$ is equal to one:

$$D_{p,q} = \sum_{k=1}^{n} | p_k - q_k | = 1$$

Definition 2 Connected path:

Let **P** ($\mathbf{P} \subseteq \mathbf{S}$) be a path which is consisted of *m* points: $p_1, \ldots, p_m$. Path **P** is connected if and only if p_k and p_{k+1} ($k \in \{1, \ldots, m-1\}$) are connected points.

Definition 3 Neighborhood point:

Let **R** (**R** $\subseteq$ **S**) be a region. A point p ($p \notin$ **R**) is neighborhood of region **R** if and only if $\exists$ another point q ($q \in$ **R**) p and q are connected points.

Definition 4 Connected region:

Let **R** (**R** $\subseteq$ **S**) be a region. **R** is a connected region if and only if $\forall\, x, y \in$ **R**, $\exists$ a connected path **P** (**P** = $\{p_1, \ldots, p_m\}$) where $p_1 = x$ and $p_n = y$.

Definition 5 Partition image:

A partition image P is a mapping P: **S** $\rightarrow$ T where T is a complete ordered lattice. Let $\mathbf{R}_p(x)$ be the region containing a point x: $\mathbf{R}_p(x) = \cup_{y \in S} \{y \mid P(x) = P(y)\}$. A partition image should satisfy the following condition: $\forall x, y \in$ **S**, $\mathbf{R}_p(x) = \mathbf{R}_p(y)$ or $\mathbf{R}_p(x) \cap \mathbf{R}_p(y) = \varnothing$; $\cup_{x \in S} \mathbf{R}_p(x) =$ **S**.

Definition 6 Connected partition image:

A connected partition image is a partition image P where $\forall\, x \in$ **S**, $\mathbf{R}_p(x)$ is always connected.

Definition 7 Fine partition:

If a partition image P is finer than another partition image P' on **S**, this means $\forall\, x \in$ **S**, $\mathbf{R}_p(x) \supseteq \mathbf{R}_{p'}(x)$.

Definition 8 Coarse partition:

If a partition image P is coarser than another partition image P' on **S**, this means $\forall\, x \in$ **S**, $\mathbf{R}_p(x) \subseteq \mathbf{R}_{p'}(x)$.

There are two extreme cases for the partition image. One is "the coarsest partition" which covers the whole S: $\forall\, x, y \in$ **S**, $\mathbf{R}_p(x) = \mathbf{R}_p(y)$. The other is called "the finest partition" where each point in **S** is an individual region: $\forall\, x, y \in$ **S**, $x \neq y \Rightarrow \mathbf{R}_p(x) \neq \mathbf{R}_p(y)$.

Definition 9 Adjacent regions:

Two regions $\mathbf{R}_1$ and $\mathbf{R}_2$ are adjacent if and only if $\exists\, x, y\, (x \in \mathbf{R}_1$ and $y \in \mathbf{R}_2)$ x and y are connected points.

Definition 10 Region adjacency graph:

Let P be a partition image on a multidimensional set **S**. There are k regions $(\mathbf{R}_1, \ldots, \mathbf{R}_k)$ in P where $\mathbf{S} = \cup\, \mathbf{R}_i$ and if $i \neq j \Rightarrow \mathbf{R}_i \cap \mathbf{R}_j = \varnothing$. The region adjacency graph (RAG) consists of a set of vertices **V** and an edge set **L**. Let $\mathbf{V} = \{v_1, \ldots, v_k\}$ where each v_i is associated to the correspondent region $\mathbf{R}_i$. The edge set **L** is $\{e_1, \ldots, e_t\}$, $\mathbf{L} \subseteq \mathbf{V} \otimes \mathbf{V}$ where each e_i is built between two vertices if the two correspondent regions are adjacent regions.

Figures 3A-C illustrate examples of different types of partition images, and Figure 3D shows an example of a region adjacency graph based on these partition images. In these examples, S is a set of two-dimensional images. The white areas 300-308, hatched areas 310-314, and spotted area 316 represent different regions in a two-dimensional image frame. Fig. 3A shows a partition image having two disconnected regions (white areas 300 and 302). Figure 3B shows a connected partition image having two connected regions (white area 304 and hatched area 312). Figure 3C shows a finer partition image as compared to Figure 3A in that hatched area 310 of Figure 3A comprises two regions: hatched area 314 and spotted area 316. Figure 3D shows the corresponding region adjacency graph of the partition image in Fig. 3C. The vertices 320, 322, 324, 326 in the graph correspond to regions 306, 314, 316, and 308, respectively. The edges 330, 332, 334, 336, and 338 connect vertices of adjacent regions.

Definition 11 Vector image sequence:

Given m ($m \geq 1$) totally ordered complete lattices $L_1, \ldots, L_m$ of product $\mathbf{L}$ ($\mathbf{L} = L_1 \otimes L_2 \otimes \ldots \otimes L_m$), a vector image sequence is a sequence of mapping $\mathbf{I}_t : \mathbf{S} \rightarrow \mathbf{L}$, where $\mathbf{S}$ is a n-dimensional set and t is in the time domain.

Several types of vector image sequences are illustrated above in Table 1. These vector image sequences can be obtained either from a series of sensors, e.g. color images, or from a computed parameter space, e.g. dense motion fields. Although the physical meaning of the input signals varies from case to case, all of them can be universally regarded as vector image sequences.

Definition 12 Semantic video objects:

Let $\mathbf{I}$ be a vector image on a n-dimensional set $\mathbf{S}$. Let P be a semantic partition image of $\mathbf{I}$. $\mathbf{S} = \cup_{i=1,\ldots,m} \mathbf{O}_i$. Each $\mathbf{O}_i$ indicates the location of a semantic video object.

Definition 13 Semantic video object segmentation:

Let $\mathbf{I}$ be a vector image on a n-dimensional set $\mathbf{S}$. Semantic video object segmentation is to find the object number m and the location of each object $\mathbf{O}_i$, $i = 1, \ldots, m$, where $\mathbf{S} = \cup_{i=1,\ldots,m} \mathbf{O}_i$.

Definition 14 Semantic video object tracking:

Let $\mathbf{I}_{t-1}$ be a vector image on a n-dimensional set $\mathbf{S}$ and P_{t-1} be the corresponding semantic partition image at time $t - 1$. $\mathbf{S} = \cup_{i=1,\ldots,m} \mathbf{O}_{t-1,i}$. Each $\mathbf{O}_{t-1,i}$ ($i = 1, \ldots, m$) is a semantic video object at time t-1. Semantic video object tracking in $\mathbf{I}_t$ is defined as finding the semantic video object $\mathbf{O}_{t,i}$ at time t, $i = 1, \ldots, m$. $\forall x \in \mathbf{O}_{t-1,i}$ and $\forall y \in \mathbf{O}_{t,i}$: $P_{t-1}(x) = P_t(y)$.

Example Implementation

The following sections describe a specific implementation of a semantic video object tracking method in more detail. Figure 4 is a block diagram illustrating the principal components in the implementation described below. Each of the blocks in

Figure 4 represent program modules that implement parts of the object tracking method outlined above. Depending on a variety of considerations, such as cost, performance and design complexity, each of these modules may be implemented in digital logic circuitry as well.

Using the notation defined above, the tracking method shown in Fig. 4 takes as input the segmentation result of a previous frame at time t-1 and the current vector image $\mathbf{I}_t$. The current vector image is defined in m ($m \geq 1$) totally ordered complete lattices $L_1, \ldots, L_m$ of product $\mathbf{L}$ (see Definition 11) on a n-dimensional set $\mathbf{S}$: $\forall p, p \in \mathbf{S}, \mathbf{I}_t(p) = \{L_1(p), L_2(p), \ldots, L_m(p)\}$.

Using this information, the tracking method computes a partition image for each frame in the sequence. The result of the segmentation is a mask identifying the position of each semantic object in each frame. Each mask has an object number identifying which object it corresponds to in each frame.

For example, consider a color image sequence as defined in Table 1. Each point p represents a pixel in a two-dimensional image. The number of points in the set S corresponds to the number of pixels in each image frame. The lattice at each pixel comprises three sample values, corresponding to Red, Green and Blue intensity values. The result of the tracking method is a series of two-dimensional masks identifying the position of all of the pixels that form part of the corresponding semantic video object for each frame.

Region Pre-Processing

The implementation shown in Fig. 4 begins processing for a frame by simplifying the input vector image. In particular, a simplifying filter 420 cleans the entire input vector image before further processing. In designing this pre-processing stage, it is preferable to select a simplifying method that does not introduce spurious data. For instance, a low pass filter may clean and smooth an image, but may also make the boundaries of a video object distorted. Therefore, it is preferable to select a method that simplifies the input vector image while preserving the boundary position of the semantic video object.

Many non-linear filters, such as median filters or morphological filters, are candidates for this task. The current implementation uses a vector median filter, *Median*(•), for the simplification of the input vector image.

The vector median filter computes the median image value(s) of neighboring points for each point in the input image and replaces the image value at the point with the median value(s). For every point p in the n-dimensional set **S**, a structure element **E** is defined around it which contains all the connected points (see Definition 1 about connected points):

$$\mathbf{E} = \cup_{q \in S} \{D_{p,q} = 1\}$$

The vector median of a point p is defined as the median of each component within the structure element **E**:

$$Median(I_t(p)) = \left\{ \underset{q \in E}{median}\{L_1(q)\}, \ ..., \ \underset{q \in E}{median}\{L_m(q)\} \right\}$$

By using such a vector median filter, small variation of the vector image $\mathbf{I}_t$ can be removed while the boundaries of video objects are well preserved under the special design of the structure element **E**. As a result, the tracking process can more effectively identify boundaries of semantic video objects.

Region Extraction

After filtering the vector input image, the tracking process extracts regions from the current image. To accomplish this, the tracking process employs a spatial segmentation method 422 that takes the current image and identifies regions of connected points having "homogenous" image values. These connected regions are the regions of points that are used in region based motion estimation 424 and region-based classification 426.

In implementing a region extraction stage, there are three primary issues to address. First, the concept of "homogeneous" needs to be consolidated. Second, the total number of regions should be found. Third, the location of each region must be

fixed. The literature relating to segmentation of vector image data describes a variety of spatial segmentation methods. Most common spatial segmentation methods use:

- polynomial functions to define the homogeneity of the regions;
- deterministic methods to find the number of regions; and/or
- boundary adjustment to finalize the location of all the regions.

These methods may provide satisfactory results in some applications, but they do not guarantee an accurate result for a wide variety of semantic video objects with non-rigid motion, disconnected regions and multiple colors. The required accuracy of the spatial segmentation method is quite high because the accuracy with which the semantic objects can be classified is dependent upon the accuracy of the regions. Preferably, after the segmentation stage, no region of the semantic object should be missing, and no region should contain an area that does not belong to it. Since the boundaries of the semantic video objects in the current frame are defined as a subset of all the boundaries of these connected regions, their accuracy directly impacts the accuracy of the result of the tracking process. If the boundaries are incorrect, then the boundary of the resulting semantic video object will be incorrect as well. Therefore, the spatial segmentation method should provide an accurate spatial partition image for the current frame.

The current implementation of the tracking method uses a novel and fast spatial segmentation method, called **LabelMinMax**. This particular approach grows one region at a time in a sequential fashion. This approach is unlike other parallel region growing processes that require all seeds to be specified before region growing proceeds from any seed. The sequential region growing method extracts one region after another. It allows more flexible treatment of each region and reduces the overall computational complexity.

The region homogeneity is controlled by the difference between the maximum and minimum values in a region. Assume that the input vector image $\mathbf{I}_t$ is defined in m ($m \geq 1$) totally ordered complete lattices $L_1, \ldots, L_m$ of product $\mathbf{L}$ (see Definition 11):

$$\forall p, p \in \mathbf{S}, \mathbf{I}_i(p) = \{L_1(p), L_2(p), \ldots, L_m(p)\}.$$

The maximum and minimum values (**MaxL** and **MinL**) in a region **R** are defined as:

$$\text{MaxL} = \left\{ \max_{p \in R}\{L_1(p)\} \ ,\ldots,\ \max_{p \in R}\{L_m(p)\} \right\}; \quad \text{MinL} = \left\{ \min_{p \in R}\{L_1(p)\} \ ,\ldots,\ \min_{p \in R}\{L_m(p)\} \right\};$$

If the difference between **MaxL** and **MinL** is smaller than a threshold (**H** = $\{h_1, h_2, \ldots, h_m\}$) that region is homogeneous:

$$\text{Homogeneity}: \forall i, 1 \le i \le m, (\max_{p \in R}\{L_i(p)\} - \min_{p \in R}\{L_i(p)\}) \le h_i$$

The **LabelMinMax** method labels each region one after another. It starts with a point *p* in the n-dimensional set **S**. Assume region **R** is the current region that **LabelMinMax** is operating on. At the beginning, it only contains the point *p*: **R** = {*p*}. Next, **LabelMinMax** checks all of the neighborhood points of region **R** (see Definition 3) to see whether region **R** is still homogeneous if a neighborhood point *q* is inserted into it. A point *q* is added into region **R** if the insertion does not change the homogeneity of that region. The point *q* should be deleted from set **S** when it is added into region **R**. Gradually, region **R** expands to all the homogeneous territories where no more neighborhood points can be added. Then, a new region is constructed with a point from the remaining points in **S**. This process continues until there are no more points left in **S**. The whole process can be clearly described by the following pseudo-code:

LabelMinMax:

```
NumberOfRegion = 0;
While (S ≠ empty) {
     Take a point p from S: R = {p}; S = S ⨳ {p};
     NumberOfRegion = NumberOfRegion + 1;
     For all the points q in S {
          if (q is in neighborhood of R) {
               if ((R + {q}) is homogeneous) {
                    R = R + {q};
                    S = S ⨳ {q};
               }
          }
     }
     Assign a label to region R, e.g. NumberOfRegion.
}
```

LabelMinMax has a number of advantages, including:

- **MaxL** and **MinL** present a more precise description about a region's homogeneity compared to other criteria;
- The definition of homogeneity gives a more rigid control over the homogeneity of a region which leads to accurate boundaries;
- **LabelMinMax** provides reliable spatial segmentation results;
- **LabelMinMax** possesses much lower computational complexity than many other approaches.

While these advantages make LabelMinMax a good choice for spatial segmentation, it also possible to use alternative segmentation methods to identify connected regions. For example, other region growing methods use different homogeneity criteria and models of "homogenous" regions to determine whether to add points to a homogenous region. .These criteria include, for example, an intensity

threshold, where points are added to a region so long as the difference between intensity of each new point and a neighboring point in the region does not exceed a threshold. The homogeneity criteria may also be defined in terms of a mathematical function that describe how the intensity values of points in a region are allowed to vary and yet still be considered part of the connected region.

Region Based Motion Estimation

The process of region-based motion estimation 424 matches the image values in regions identified by the segmentation process with corresponding image values in the previous frame to estimate how the region has moved from the previous frame. To illustrate this process, consider the following example. Let $\mathbf{I}_{t-1}$ be the previous vector image on a n-dimensional set **S** at time t-1 and let $\mathbf{I}_t$ be the current vector image on the same set **S** at time t. The region extraction procedure has extracted N homogeneous regions $\mathbf{R}_i$ (i = 1, 2, ..., N) in the current frame $\mathbf{I}_t$:

$$\mathbf{S} = \cup_{i=1,\ldots,N} \mathbf{R}_i.$$

Now, the tracking process proceeds to classify each region as belonging to exactly one of the semantic video objects in the previous frame. The tracking process solves this region-based classification problem using region-based motion estimation and compensation. For each extracted region $\mathbf{R}_i$ in the current frame $\mathbf{I}_t$, a motion estimation procedure is carried out to find where this region originates in the previous frame $\mathbf{I}_{t-1}$. While a number of motion models may be used, the current implementation uses a translational motion model for the motion estimation procedure. In this model, the motion estimation procedure computes a motion vector V_i for region $\mathbf{R}_i$ that minimizes the prediction error (**PE**) on that region:

$$\mathrm{PE} = \min_{V_i}\left\{\sum_{p\in R_i} \| I_t(p) - I_{t-1}(p+V_i)\|\right\}$$

where $\|\bullet\|$ denotes the sum of absolute difference between two vectors and $V_i \leq V_{max}$ (V_{max} is the maximum search range). This motion vector V_i is assigned to region $\mathbf{R}_i$ indicating its trajectory location in the previous frame $\mathbf{I}_{t-1}$.

Other motion models may be used as well. For example, an affine or perspective motion model can be used to model the motion between a region in the current vector image and a corresponding region in the previous vector image. The affine and perspective motion models use a geometric transform (e.g., an affine or perspective transform) to define the motion of region between one frame and another. The transform is expressed in terms of motion coefficients that may be computed by finding motion vectors for several points in a region and then solving a simultaneous set of equations using the motion vectors at the selected points to compute the coefficients. Another way is to select an initial set of motion coefficients and then iterate until the error (e.g., a sum of absolute differences or a sum of squared differences) is below a threshold.

Region Based Classification

The region based classification process 426 modifies the location of each region using its motion information to determine the region's estimated position in the previous frame. It then compares this estimated position with the boundaries of semantic video objects in the previous frame ($\mathbf{S}_t$) to determine which semantic video object that it most likely forms a part of.

To illustrate, consider the following example. Let $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$ be the previous and current vector images on a n-dimensional set $\mathbf{S}$ and P_{t-1} be the corresponding semantic partition image at time t-1:

$$\mathbf{S} = \cup_{i=1,\ldots,m} \mathbf{O}_{t-1,i}.$$

Each $\mathbf{O}_{t-1,i}$ ($i = 1, \ldots, m$) indicates the location of a semantic video object at time t-1. Assume that there are N total extracted regions $\mathbf{R}_i$ ($i = 1, 2, \ldots, N$) , each having an associated motion vector V_i ($i = 1, 2, \ldots, N$) in the current frame. Now, the tracking method needs to construct the current semantic partition image P_t at the time t.

The tracking process fulfils this task by finding a semantic video object $\mathbf{O}_{t-1,j}$ ($j \in \{1, 2, \ldots, m\}$) for each region $\mathbf{R}_i$ in the current frame.

Since the motion information for each region $\mathbf{R}_i$ is already available at this stage, the region classifier 426 uses backward motion compensation to warp each

region $\mathbf{R}_i$ in the current frame towards the previous frame. It warps the region by applying the motion information for the region to the points in the region. Let's assume the warped region in the previous frame is $\mathbf{R}'_i$:

$$\mathbf{R}'_i = \cup_{p \in Ri} \{p + V_i\}.$$

Ideally, the warped region $\mathbf{R}'_i$ should fall onto one of the semantic video objects in the previous frame:

$$\exists j, j \in \{1, 2, \ldots, m\} \text{ and } \mathbf{R}'_i \subseteq \mathbf{O}_{t-1,j}.$$

If this is the case, then the tracking method assigns the semantic video object $\mathbf{O}_{t-1,j}$ to this region $\mathbf{R}_i$. However, in reality, because of the potentially ambiguous results from the motion estimation process, $\mathbf{R}'_i$ may overlap with more than one semantic video object in the previous frame, i.e.

$$\mathbf{R}'_i \not\subset \mathbf{O}_{t-1,j}, j = 1, 2, \ldots, m.$$

The current implementation uses **majority** criteria M for the region-based classification. For each region $\mathbf{R}_i$ in the current frame, if the majority part of the warped region $\mathbf{R}'_i$ comes from a semantic video object $\mathbf{O}_{t-1,j}$ ($j \in \{1, 2, \ldots, m\}$) in the previous frame, this region is assigned to that semantic video object $\mathbf{O}_{t-1,j}$:

$$\forall p \in \mathbf{R}_i \text{ and } \forall q \in \mathbf{O}_{t-1,j}, P_t(p) = P_{t-1}(q).$$

More specifically, the semantic video object $\mathbf{O}_{t-1,j}$ that has the majority overlapped area (MOA) with $\mathbf{R}'_i$ is found as:

$$M: \text{MOA} = \max_j \left\{ \sum_{p \in Ri} N_j(p + V_i), j = 1, \ldots, m \right\}; \quad N_j(p + V_i) = \begin{cases} 1 & (p + V_i) \in O_{t-1,j} \\ 0 & (p + V_i) \notin O_{t-1,j} \end{cases}$$

Piece by piece, the complete semantic video objects $\mathbf{O}_{t,j}$ in the current frame are constructed using this region-based classification procedure for all the regions $\mathbf{R}_i$ ($i = 1, 2, \ldots, N$) in the current frame. Assume a point $q \in \mathbf{O}_{t-1,j}$,

$$\mathbf{O}_{t,j} = \cup_{p \in S} \{p \mid P_t(p) = P_{t-1}(q)\}, j = 1, 2, \ldots, m.$$

According to the design of this region-based classification process, there will not be any holes/gaps or overlaps between different semantic video objects in the current frame:

$$\cup_{i=1,\ldots,m} \mathbf{O}_{t,i} = \cup_{i=1,\ldots,N} \mathbf{R}_i = \cup_{i=1,\ldots,m} \mathbf{O}_{t-1,i} = \mathbf{S}.$$

$$\forall i, j \in \{1, \ldots, m\}, i \neq j \Rightarrow \mathbf{O}_{t,i} \cap \mathbf{O}_{t,j} = \varnothing;$$

This is an advantage of the tracking system compared to tracking systems that track objects into frames where the semantic video object boundaries are not determined. For example, in forward tracking systems, object tracking proceeds into subsequent frames where precise boundaries are not known. The boundaries are then adjusted to fit an unknown boundary based on some predetermined criteria that models a boundary condition.

Region Post-Processing

Let's assume the tracking result in the current frame is the semantic partition image P_t. For various reasons, there might be some errors in the region-based classification procedure. The goal of the region post-processing process is to remove those errors and at the same time to smooth the boundaries of each semantic video object in the current frame. Interestingly, the partition image is a special image that is different from the traditional ones. The value in each point of this partition image only indicates the location of a semantic video object. Therefore, all the traditional linear or non-linear filters for signal processing are not generally suitable for this special post-processing.

The implementation uses a **majority** operator $M(\bullet)$ to fulfil this task. For every point p in the n-dimensional set **S**, a structure element **E** is defined around it which contains all the connected points (see 1 about connected points):

$$\mathbf{E} = \cup_{q \in S} \{D_{p,q} = 1\}$$

First, the **majority** operator $M(\bullet)$ finds a semantic video object $\mathbf{O}_{t,j}$ which has the maximal overlapped area (MOA) with the structure element **E**:

$$\mathrm{MOA} = \max_j \left\{ \sum_{q \in \mathbf{E}} \mathrm{N}_j(q), j = 1, \ldots, m \right\}; \quad \mathrm{N}_j(q) = \begin{cases} 1 & q \in \mathrm{O}_{t,j} \\ 0 & q \notin \mathrm{O}_{t,j} \end{cases}$$

Second, the **majority** operator $M(\bullet)$ assigns the value of that semantic video object $\mathbf{O}_{t,j}$ to the point p:

$$\text{Let } q \in \mathbf{O}_{i,j},\ P_i(p) = M(p) = P_i(q).$$

Because of the adoption of the majority criteria, very small areas (which most likely are errors) may be removed while the boundaries of each semantic video object are smoothed.

Brief Overview of a Computer System

Figure 5 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although the invention or aspects of it may be implemented in a hardware device, the tracking system described above is implemented in computer-executable instructions organized in program modules. The program modules include the routines, programs, objects, components, and data structures that perform the tasks and implement the data types described above.

While Fig. 5 shows a typical configuration of a desktop computer, the invention may be implemented in other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be used in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Figure 5 illustrates an example of a computer system that serves as an operating environment for the invention. The computer system includes a personal computer 520, including a processing unit 521, a system memory 522, and a system bus 523 that interconnects various system components including the system memory to the processing unit 521. The system bus may comprise any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using a bus architecture such as PCI, VESA, Microchannel (MCA), ISA and EISA, to name a few. The system memory includes read only memory (ROM) 524 and random access memory (RAM) 525. A basic input/output system 526 (BIOS),

containing the basic routines that help to transfer information between elements within the personal computer 520, such as during start-up, is stored in ROM 524. The personal computer 520 further includes a hard disk drive 527, a magnetic disk drive 528, e.g., to read from or write to a removable disk 529, and an optical disk drive 530, e.g., for reading a CD-ROM disk 531 or to read from or write to other optical media. The hard disk drive 527, magnetic disk drive 528, and optical disk drive 530 are connected to the system bus 523 by a hard disk drive interface 532, a magnetic disk drive interface 533, and an optical drive interface 534, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions (program code such as dynamic link libraries, and executable files), etc. for the personal computer 520. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD, it can also include other types of media that are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like.

A number of program modules may be stored in the drives and RAM 525, including an operating system 535, one or more application programs 536, other program modules 537, and program data 538. A user may enter commands and information into the personal computer 520 through a keyboard 540 and pointing device, such as a mouse 542. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 521 through a serial port interface 546 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 547 or other type of display device is also connected to the system bus 523 via an interface, such as a display controller or video adapter 548. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 520 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 549.

The remote computer 549 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 520, although only a memory storage device 50 has been illustrated in Figure 5. The logical connections depicted in Figure 5 include a local area network (LAN) 551 and a wide area network (WAN) 552. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 520 is connected to the local network 551 through a network interface or adapter 553. When used in a WAN networking environment, the personal computer 520 typically includes a modem 554 or other means for establishing communications over the wide area network 552, such as the Internet. The modem 554, which may be internal or external, is connected to the system bus 523 via the serial port interface 546. In a networked environment, program modules depicted relative to the personal computer 520, or portions thereof, may be stored in the remote memory storage device. The network connections shown are merely examples and other means of establishing a communications link between the computers may be used.

Conclusion

While the invention is described in the context of specific implementation details, it is not limited to these specific details. The invention provides a semantic object tracking method and system that identifies homogenous regions in a vector image frame and then classifies these regions as being part of a semantic object. The classification method of the implementation described above is referred to as "backward tracking" because it projects a segmented region into a previous frame where the semantic object boundaries are previously computed.

Note that this tracking method also generally applies to applications where the segmented regions are projected into frames where the semantic video object boundaries are known, even if these frames are not previous frames in an ordered sequence. Thus, the "backward" tracking scheme described above extends to

applications where classification is not necessarily limited to a previous frame, but instead to frames where the semantic object boundaries are known or previously computed. The frame for which semantic video objects have already been identified is more generally referred to as the reference frame. The tracking of the semantic objects for the current frame are computed by classifying segmented regions in the current frame with respect to the semantic object boundaries in the reference frame.

As noted above, the object tracking method applies generally to vector image sequences. Thus, it is not limited to 2D video sequences or sequences where the image values represent intensity values.

The description of the region segmentation stage identified criteria that are particularly useful but not required for all implementations of semantic video object tracking. As noted, other segmentation techniques may be used to identify connected regions of points. The definition of a region's homogeneity may differ depending on the type of image values (e.g., motion vectors, color intensities, etc.) and the application.

The motion model used to perform motion estimation and compensation can vary as well. Though computationally more complex, motion vectors may be computed for each individual point in a region. Alternatively, a single motion vector may be computed for each region, such as in the translational model described above. Preferably, a region based matching method should be used to find matching regions in the frame of interest. In region based matching, the boundary or mask of the region in the current frame is used to exclude points located outside the region from the process of minimizing the error between the predicted region and corresponding region in the target frame. This type of approach is described in co-pending U.S. Patent Application No. 08/657,274, by Ming-Chieh Lee, entitled Polygon Block Matching Method, which is hereby incorporated by reference.

In view of the many possible implementations of the invention, the implementation described above is only an example of the invention and should not be taken as a limitation on the scope of the invention. Rather, the scope of the invention

is defined by the following claims. We therefore claim as our invention all that comes within the scope and spirit of these claims.